归并排序

归并排序

- 将数组A[0...n-1]分为近似相等的子数组，并将其分别拷贝到数组B和C
- 对数组B和C分别递归排序
- 按如下规则对B和C进行归并:
  - 重复下述过程直至一个数组为空:
    - 将两个数组未处理部分的第一个元素进行对比
    - 将两个数中的较小者拷贝至A, 并同时增加指示该数组未处理部分的索引
  - 一旦一个数组中所有元素都被排序完毕，直接将另一数组的剩余元素复制到数组A

归并排序(伪代码)

排序过程：

ALGORITHM *Mergesort*(*A*[0...n-1])

if n > 1

copy $A[0...\lfloor n/2 \rfloor\text{-}1]$ to $B[0...\lfloor n/2 \rfloor\text{-}1]$

copy $A[\lfloor n/2 \rfloor...n\text{-}1]$ to $C[0...\lceil n/2 \rceil\text{-}1]$

Mergesort($B[0...\lfloor n/2 \rfloor\text{-}1]$)

Mergesort($C[0...\lceil n/2 \rceil\text{-}1]$)

Merge(B, C, A)

需要长度为n的辅助数组，空间复杂度$O(n)$

归并过程：

ALGORITHM *Merge*(B[0...p-1], $C[0...q-1]$, $A[0...p+q-1]$)

$i \leftarrow 0$; j$\leftarrow 0$; $k \leftarrow 0$

while $i < p$ **and** $j < q$ **do**

if $B[i] \le C[j]$

$A[k] \leftarrow B[i]$; $i \leftarrow i+1$

else $A[k] \leftarrow C[i]$; $j \leftarrow j+1$

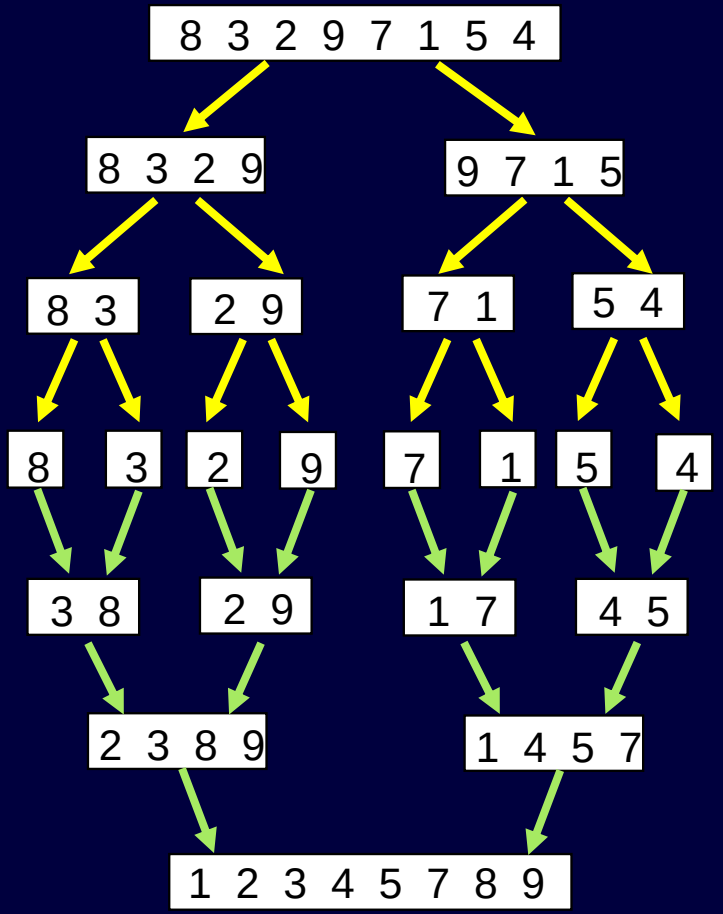
$k \leftarrow k+1$

if $i = p$

copy $C[j...q-1]$ to $A[k...p+q-1]$)

else *copy* $B[i...q-1]$ to $A[k...p+q-1]$)

归并排序(示例)



- 归并排序总时间=分解时间+子序列排序时间+归并时间
- 每个拆分都是折中分解，所以分解时间是个常数，可以忽略不计。
- 假设长度为n的数组归并排序时间为T(n)

归并排序总时间=子序列排序时间+归并时间

↓

T(n) = 2*T(n/2)+1*n （第2层）

= 2*(2*T(n/4)+n/2)+n = 4*T(n/4)+2*n

= … = n*T(1)+($\log_2^n$)*n （第$\log_2^n$+1层）

归并排序的时间复杂度为$O(nlgn)$

自底向上的归并排序

- 自底向上的归并排序直接将原数组切分为n个长度为1的数组，对其进行两两归并，将归并后得到的n/2个数组继续两两归并，直至归并结束。这种排序方法不需要递归，并且不需要辅助空间。
- 如果n为奇数，则将最后一个子数组轮空，将前n-1个子数组进行归并

ALGORITHM *Mergesort _ Bottom2Top*$(A[0...n-1])$

```
size ← 1;      //从size为1开始归并
while size < length do
    left ← 0;
    while left < n - size do
        if left + 2*size -1 < n -1 do
            merge(A[left...left + size -1], A[left + size...left + 2*size -1], A[0...n -1]);
        else   merge(A[left...left + size -1], A[left + size...n -1], A[0...n -1]);
        left ← left + 2*size;
    size ← 2*size;   //size翻倍
```